Employee Data :

[cloudera@quickstart pig]$ cat employee\_details.txt

101,Amitabh,20000,1

102,Shahrukh,10000,2

103,Akshay,11000,3

104,Anubhav,5000,4

105,Pawan,2500,5

106,Aamir,25000,1

107,Salman,17500,2

108,Ranbir,14000,3

109,Katrina,1000,4

110,Priyanka,2000,5

111,Tushar,500,1

112,Ajay,5000,2

113,Jubeen,1000,1

114,Madhuri,2000,2[cloudera@quickstart pig]$

1. Top 5 employees (employee id and employee name) with highest rating. (In case two employees have same rating, employee with name coming first in dictionary should get preference)

emp = LOAD 'employee\_details.txt' USING PigStorage(',') AS (emp\_id:int, emp\_name:chararray, emp\_salary:int, rank:int) ;

grunt> describe emp ;

emp: {emp\_id: int,emp\_name: chararray,emp\_salary: int,rank: int}

sorted\_emp\_by\_rank = ORDER emp by rank ;

dump sorted\_emp\_by\_rank ;

Output:

ne.util.MapRedUtil - Total input paths to process : 1

(106,Aamir,25000,1)

(101,Amitabh,20000,1)

(113,Jubeen,1000,1)

(111,Tushar,500,1)

(114,Madhuri,2000,2)

(112,Ajay,5000,2)

(102,Shahrukh,10000,2)

(107,Salman,17500,2)

(103,Akshay,11000,3)

(108,Ranbir,14000,3)

(104,Anubhav,5000,4)

(109,Katrina,1000,4)

(110,Priyanka,2000,5)

(105,Pawan,2500,5)

top5= limit sorted\_emp\_by\_rank 5 ;

OUTPUT:

2018-05-19 17:00:05,754 [main] INFO org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1

2018-05-19 17:00:05,754 [main] INFO org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1

(111,Tushar,500,1)

(113,Jubeen,1000,1)

(101,Amitabh,20000,1)

(106,Aamir,25000,1)

(114,Madhuri,2000,2)

(b) Top 3 employees (employee id and employee name) with highest salary, whose employee id

is an odd number. (In case two employees have same salary, employee with name coming first

in dictionary should get preference)

(c) Employee (employee id and employee name) with maximum expense (In case two

employees have same expense, employee with name coming first in dictionary should get

preference)

emp\_expenses = LOAD 'employee\_expenses' AS (emp\_id:int, expenses:int);

dump emp\_expenses;

OUTPUT :

ne.util.MapRedUtil - Total input paths to process : 1

(101,200)

(102,100)

(110,400)

(114,200)

(119,200)

(105,100)

(101,100)

(104,300)

(102,400)

grunt> joined\_data = join emp by emp\_id , emp\_expenses by emp\_id;

grunt> dump joined\_data;

OUTPUT :

(101,Amitabh,20000,1,101,100)

(101,Amitabh,20000,1,101,200)

(102,Shahrukh,10000,2,102,400)

(102,Shahrukh,10000,2,102,100)

(104,Anubhav,5000,4,104,300)

(105,Pawan,2500,5,105,100)

(110,Priyanka,2000,5,110,400)

(114,Madhuri,2000,2,114,200)

grunt> describe joined\_data ;

joined\_data: {emp::emp\_id: int,emp::emp\_name: chararray,emp::emp\_salary: int,emp::rank: int,emp\_expenses::emp\_id: int,emp\_expenses::expenses: int}

grunt> joined\_data\_by\_expense= order joined\_data by expenses desc ;

Output :

(110,Priyanka,2000,5,110,400)

(102,Shahrukh,10000,2,102,400)

(104,Anubhav,5000,4,104,300)

(114,Madhuri,2000,2,114,200)

(101,Amitabh,20000,1,101,200)

(105,Pawan,2500,5,105,100)

(102,Shahrukh,10000,2,102,100)

(101,Amitabh,20000,1,101,100)

top\_expenses\_1 = limit joined\_data\_by\_expense 1 ;

OUTPUT :

(110,Priyanka,2000,5,110,400)

(d) List of employees (employee id and employee name) having entries in employee\_expenses

file.

joined\_data = join emp by emp\_id , emp\_expenses by emp\_id;

OUTPUT :

(101,Amitabh,20000,1,101,100)

(101,Amitabh,20000,1,101,200)

(102,Shahrukh,10000,2,102,400)

(102,Shahrukh,10000,2,102,100)

(104,Anubhav,5000,4,104,300)

(105,Pawan,2500,5,105,100)

(110,Priyanka,2000,5,110,400)

(114,Madhuri,2000,2,114,200)

Name= distinct ( FOREACH joined\_data generate emp\_name) ;

(Pawan)

(Amitabh)

(Anubhav)

(Madhuri)

(Priyanka)

(Shahrukh)

(e) List of employees (employee id and employee name) having no entry in employee\_expenses

file.

joined\_data = join emp by emp\_id left outer, emp\_expenses by emp\_id;

(101,Amitabh,20000,1,101,100)

(101,Amitabh,20000,1,101,200)

(102,Shahrukh,10000,2,102,400)

(102,Shahrukh,10000,2,102,100)

(103,Akshay,11000,3,,)

(104,Anubhav,5000,4,104,300)

(105,Pawan,2500,5,105,100)

(106,Aamir,25000,1,,)

(107,Salman,17500,2,,)

(108,Ranbir,14000,3,,)

(109,Katrina,1000,4,,)

(110,Priyanka,2000,5,110,400)

(111,Tushar,500,1,,)

(112,Ajay,5000,2,,)

(113,Jubeen,1000,1,,)

(114,Madhuri,2000,2,114,200)

grunt> Filtered\_data= filter joined\_data by expenses is NULL ;

dump Filtered\_data;

output :

(103,Akshay,11000,3,,)

(106,Aamir,25000,1,,)

(107,Salman,17500,2,,)

(108,Ranbir,14000,3,,)

(109,Katrina,1000,4,,)

(111,Tushar,500,1,,)

(112,Ajay,5000,2,,)

(113,Jubeen,1000,1,,)